**Project 4**

**Design and Implement Software System**

**Mission:**

Your manager told you that the customer changed their mind and now they want you to design and implement a simulated hand game based on the “Rock, Paper, Scissors, Saw” game using Object Oriented approach and Java. The function requirements documented in SRS-RPSS-Game-OO.docx are the same but the design is different. You must use Object Oriented approach and Java for this stand-alone PC program.

**Objective:**

The goal of this project is to use basic requirements to practice creating a design (solution to the problem) and then implement that design. As part of the project you are to design interfaces and other aspects of the game, practice modularity and **object oriented approach** to implementation, and practice for the implementation to be exhibiting good practice techniques and meet both the requirements and the design.

**Design Approach:**

You are free to design the look and feel of the game as you want as long as you meet all the basic requirements given in SRS-RPSS-Game.docx. You must design and implement the game as it is described in those requirements and NOT as the original game is played or some other rules you might know. Note that the requirements document has been updated in **section 2.2 to indicate what classes you must implement**.

I will not be grading on creating a fancy design. I will be looking for correct Object-Oriented approach with realistic interfaces and reasonable class/method structure for the system. You want your design to be as complete and detailed as you can. Everything the customer has provided in the requirements document must make it into the correct sections of the design document.

You do not however have to make it GUI based. Text based interfaces are perfectly fine. I am not testing you on programming skills but rather that you can design a system to meet the requirements and then code to meet that design. Obviously you must have the basic programming skills in order to write the code but you do not have to make it fancy.

**Implementation Approach:**

As far as your code, it obviously must work and work correctly to meet all the requirements above and the game itself and the design you create. In addition, you need to make sure to use the good coding guidelines (e.g. layout, modularity, comments, etc.) that you have been reading about and the object oriented approach.

You will get partial credit when you have some mistakes in the code or it does not meet specification exactly HOWEVER, there are a few criteria that you must meet or you will not get any credit for the implementation. They include:

1. Your game must be Object Oriented – if you submit a program written as procedural or a single class, you will get 0 points for the implementation.
2. Your program must consist of the four classes indicated in the design section of the requirement specification document in section 2.2 and they must be implemented as described.
3. Your game must work with two human users and your program will be the last player as described in the requirements specification document. The player’s names have to be obtained and used throughout the program. If your design and/or implementation do not support this, you will receive 0 points for the implementation.
4. The design and implementation must include the menu as described in the requirements specification called from GameDriver::main(). No credit will be awarded if the design or implementation does not include the menu.
5. Your program must execute in order to get any points for it. If your program cannot be executed because of syntax and logic errors, no points will be awarded.

**Your specification document at minimum must include (use the provided template):**

* Introduction section (purpose, intended audience, product scope, references, system overview showing the basic flow and high level view of the functionality of the system)
* Design considerations section (assumptions, dependencies, constraints, and development approach)
* System Architecture - High-level view of your system’s architecture –chart showing high level objects and the interactions between them, specifically **class diagram** for OO structure (classes, attributes, and methods)
* Detailed design: For each class from the high-level view, give additional information: a short description of what it does and what other classes it interacts with. Each attribute should have a data type and what information it will store. Each method should have a description what it will be used for and what other methods it will interact with. If there is an algorithm related to a method (e.g. game flow) it should be described in English (not code or pseudo code).
* Human Interface Design section – the input/output described or shown as diagrams of all screens whether GUI or text-driven

**Resources to help with project:**

Project4-MSIT660-GradingCriteria.docx – criteria and points how the project will be graded

OOExampleSpecification.docx – a simple specification document that includes requirements and design to show you how the design is done for a problem  
OOExample.zip - code implementation of the OOExample so you can see how design maps to actual code

**Code examples:**

The self-assessment document you received the first week contains many examples of the java syntax you may find useful. They are there to help you with your design and coding for this project. You can use the design approaches and code in the examples I provide, modifying them for your particular design and game. You can also design and code your own way. I only provide these to help those new or struggling but if you know or like a different way, you are welcome to do it another way as long as you meet the instructions and guidelines I provide here.

**Deliverables:**

You must use the provided template (project4-specification-template.doc) for your design specification document. Your specification document must be in Microsoft .doc or .docx format named Your-name-Project4-Specification.docx and the working source code for the game. You can submit the source code as zip with all the java files in it. Make sure your document specifies the version of Java and any non-standard libraries that are needed to run your code.

**Plagiarism reminder:**

Make sure you write your own code. If you copy a solution from someplace and then modify it, you are plagiarizing and using someone else's work as your own. You are not submitting your original work. That is different from looking up how to do something in Java, whether input/output, a loop, a menu, or a function in a tutorial example or an example I give you and then using that syntax as a model to write your own code and solution. Or looking up some other program, unrelated to RPS, whether in a textbook or online to again, give you an idea how to code something that you designed but do not know the Java syntax for. If you found a RPS solution and copied from it, whether you modified it or not, you are plagiarizing. The exception is any examples I give you in this course. You can use as you wish in your design and programs.

**References**

<http://en.wikipedia.org/wiki/Rock_paper_scissors>